DOI:10.16526/j.cnki.11-4762/tp.2014.11.062

计算机测量与控制.2014.22(11)
Computer Measurement & Control

文章编号:1671-4598(2014)11-3747-04 中图分类号:TP391 文献标识码:A

基于GM(1,1)预测和虚拟机迁移的云计算负载均衡策略设计

王志勃[1,2],毕艳茹[2]

(1.江苏省电子产品装备与制造中心,江苏 淮安 223003;2.淮安信息职业技术学院,江苏 淮安 22300)

摘要:针对云计算中心虚拟机集群负载的不均衡问题,设计了一种基于GM(1,1)预测和虚拟机迁移的负载均衡策略;首先,描述了云计算中心的负载均衡原理,介绍了经典ELB算法并总结了其缺点,然后,设计了基于GM(1,1)的虚拟机负载预测算法,能根据虚拟机的历史负载信息来预测下一时刻的负载信息,能有效克服ELB算法中仅依赖当前时刻负载而进行虚拟机的删除和增加,同时通过设定不同的阈值来实现虚拟机的增加、删除和迁移,最后,定义了基于预测机制和虚拟机迁移的负载均衡算法,能在创建虚拟机时根据用户的请求定制AMIs模板,从而提交用户请求的响应速度;在CloudSim环境下进行实验,实验结果表明文中方法能有效地实现云计算中心虚拟机集群的负载均衡,与其它方法相比,具有负载均衡高和负载均衡效率高的优点,是一种有效的云计算环境的负载均衡策略。

关键词:灰色模型;虚拟机迁移;负载均衡;云计算;预测

Design for Load Balance Strategy of Cloud Computing Based on GM(1,1) Prediction and Virtual machine Migration

Wang Zhibo[1,2], Bi Yanru[2]

(1. Jiangsu Province Electronic Products and Manufacturing Center, Huaian 223003, China;
2. Huaian College of Information Technology, Huaian 223003, China)

Abstract: Aiming at the load unbalance problem of cloud computing virtual machine, a load balance strategy based on GM(1,1) prediction and virtual machine migration is proposed. Firstly, the load balance principle is described and the classic ELB algorithm is introduced, and the defects is summarized. Then the load balance algorithm based on GM(1,1) is designed, the load information can be predicted based on history load information. It can realize add and delete of the virtual machines relying on the current load of ELB algorithm. The threshold can be set as different values for add, delete and migration. Finally, the load balance algorithm based on prediction mechanism and virtual machine migration is defined. The experiment is operated in the CloudSim, the experiment shows the method in this paper can realize the load balance in cloud computing virtual machine, and compared with the other methods, it has high load balance and load balance efficiency, therefore, it is an effective load balance strategy in cloud computing.

Keywords: GM(1,1) model; virtual machine migration; load balance; cloud computing; prediction

0 引言

随着计算机技术的普及和发展,使得基于计算机的应用对计算和存储能力有了日益增高的要求,导致了云计算计算模式的产生[1-2]。云计算作为一种新的计算模式,能实现资源的统一管理和调度,使得用户无需对其管理[3],仅需通过按时付费地使用资源池中的服务[4-5]。虚拟化技术能实现对硬件资源的统一有效管理[6],由于虚拟资源数量多、动态变化快、负载具有不确定性以及物理服务器处理性能的差异,使得云计算中的虚拟资源调度容易出现负载不均衡[7]。

目前已有的实现虚拟机负载均衡的资源调度和迁移技术主要有:文献[8]提出了一种针对云计算环境的虚拟机动态部署模型。文献[9]设计了一种新的负载均衡算法,能根据负载的预测趋势实现虚拟机的动态增加和删除。文献[10]提出了一种基于迁移技术的虚拟机调度算法,从而实现数据中心的高效动态负载均衡。文献[11]提出了一种最小迁移代价的虚拟机放置算法,综合考虑虚拟机资源分配的持续变化和迁移代价,实现资源高效利用和服务质量提高。文献[12]根据系统负载状况自动地调整智能蚂蚁的搜索半径来进行搜索,实现服务器的自动迁移。

上述算法研究云计算环境的虚拟机调度和迁移问题,但没有充分利用云计算的弹性和按需使用,不能根据对负载的趋势预测来实现虚拟的负载均衡,从而解决云计算环境下由于虚拟机迁移的延迟性导致任务的延迟调度而导致的无法满足用请求的问题。

1 负载均衡算法

1.1 云计算负载均衡管理

Amazon提出了弹性计算云的概念后,其生产的云计算产品EC2得到广泛的应用,其主要原理为:用户根据自己的需求,将数据通过Amazon存储在云中,并采用云中的计算资源和通信资源来对数据进行处理,并在处理完毕后动态释放资

收稿日期:2014-07-24; 修回日期:2014-09-04。
基金项目:国家重点星火计划项目(2011GA690005)。
作者简介:王志勃(1970-),男,江苏淮安人,硕士,讲师,工程师,主要从事云计算和计算机应用方向的研究。

源，由于 EC2 提供了各种 Linux 版本的机器镜像（amazon machine images，AMIs），选择 AMIs 来创建虚拟机实例，当用户选择虚拟机，没有找到合适的虚拟机模板时可以自己定制 AMIs，将自己的应用安装 AMIs 中，此后，用户可以直接使用自己定制过的虚拟机。

EC2 中的负载均衡算法 ELB 主要关注当用户任务请求到来时，是否应该增加或删除虚拟机以实现负载平衡，因此，负载均衡器需要周期性地定时收集虚拟机的资源信息。

假设每隔时间 T 对虚拟机进行一次调度，每隔时间 t 获取服务器资源 i 负载数据，并将其负载信息加入服务器的负载队列 Q，其长度为 T/t，当队列已满时，新加入的负载数据会使队头的数据出队，服务器池维护所有同类服务的服务器，采用服务器池队列 $poolist$，用于存放正在进行用户任务调度的服务器池，所有的服务器池均有一个删除队列 $deleteQ$，表示已经处于删除状态的服务器虚拟机。

1.2 ELB 任务调度负载均衡算法

ELB 任务调度负载均衡算法可以描述为：

1）在每个时间周期 T 到来时，所有服务器查看其负载队列 Q 中的任务负载，如果任务负载数已满为 $|T/t|$，则计算出队列平均负载：

$$AveL = \sum_{i=1}^{n} l_i \,|\, T/t \,| \tag{1}$$

否则其 $AveL$ 为 0。

2）对于 $AveL$ 不为 0 的服务器 s_i，查找其所属于的服务器池 $pool_j$，并将 $pool_j$ 加入其所属服务器池队列 $poolist$ 中；

3）服务器池队列 $poolist$ 的队首元素 $poolh$ 出队，计算 $poolh$ 的平均负载 $Avepoolh$：

$$Avepoolh = \sum_{s_i \in poolh}^{n} s_i / \,|\, poolh \,| \tag{2}$$

4）如果 $Avepoolh$ 高于最大负载率阈值，则清空 $Avepoolh$ 中所有服务器的负载队列，使得 $Avepoolh$ 中的所有服务器在下一个时间周期 T 时间内不进行调度，然后查看其删除队列 $deleteQ$：

如果该队列中的服务器数 n_s 大于阈值 Num，则将 $deleteQ$ 中的 Num 个服务器的由删除状态更改为正常状态，否则将 n_s 个服务器全部修改为正常状态，并根据 AMIs 镜像模板新建 $Num - n_s$ 个虚拟机，并将其加入到服务器池 $poolh$ 中；

5）如果 $Avepoolh$ 小于最小负载率阈值，则清空 $Avepoolh$ 中所有服务器的负载队列，使得 $Avepoolh$ 中的所有服务器在下一个时间周期 T 时间内不进行调度，然后查看其删除队列 $deleteQ$：

如果该队列中的服务器数 n_s 大于阈值 Num，则将其中的 Num 个服务器的由正常状更改为态删除状态，并将其加入到 $deleteQ$ 中；否则将 n_s 个服务器全部修改由正常状更改为态删除状态，并将其加入到 $deleteQ$ 中；

6）如果 $Avepoolh$ 大于最小负载率阈值小于最大负载率阈值，则不进行虚拟机的删除或增加操作。

2 基于 GM（1，1）的负载预测

2.1 负载预测原理

ELB 算法中仅简单地根据当前时间周期 T 中虚拟机的负载情况进行采集，没有对历时负载数据进行存储，因此，无法通过对历时负载数据和当前运行状态信息的综合分析，实现对下一个周期的负载情况进行预测，从而有效地确定下一步的动作（增加虚拟机、删除虚拟机、迁移虚拟机）。由于创建、删除和迁移虚拟机都会造成一定的时空开销，从而会造成一定的性能损耗，严重时甚至导致用户请求得不到及时响应。

因此，文中采用基于 GM（1，1）的预测机制实现对虚拟机负载的预测。灰色模型 GM（1，1）是一种针对小样本的能处理不确定性情况的较为精确的预测模型，仅需要少量历史数据信息就能对下一时刻的数据进行有效精确地预测。

2.2 基于 GM（1，1）的预测算法

为了实现对负载进行预测，负载均衡器中需要保存长度为 n 周期为 T 的历史负载信息，并对所有的服务器池，均通过下列算法预测在下一个时间周期 T 内服务器池中的服务器数量：

算法 1：基于 GM（1，1）的虚拟机负载预测算法。

1）将长度为 n 周期为 T 的历史负载信息在 T_0 周期时的值表示为：

$$X^{(0)} = \{X^{(0)}(1), X^{(0)}(2), \ldots, X^{(0)}(n)\} \tag{3}$$

2）对式（3）所示的 $X^{(0)}$ 进行一次累加生操作：

$$X^{(1)} = \{X^{(1)}(1), X^{(1)}(2), \ldots, X^{(1)}(n)\} \tag{4}$$

其中，$X^{(1)}(k)$ 可以根据下式获取：

$$X^{(1)}(k) = \sum_{i=1}^{k} X^{(0)}(k), k = 1, 2, \ldots, n \tag{5}$$

3）通过 $X^{(0)}(k)$ 与 $X^{(1)}(k-1)$ 的比值 $\rho(k)$ 对 $X^{(0)}$ 进行光滑检验，并根据 $X^{(1)}(k)$ 与 $X^{(1)}(k-1)$ 的比值 $\sigma^{(1)}(k)$ 判断 $X^{(1)}(k)$ 是否满足指数规律：

$$\rho(k) = \frac{X^{(0)}(k)}{X^{(1)}(k-1)} \tag{6}$$

$$\sigma^{(1)}(k) = \frac{X^{(1)}(k)}{X^{(1)}(k-1)} \tag{7}$$

4）生成 $X^{(1)}(k)$ 的紧邻域 $Z^{(1)}(k)$，如下所示：

$$Z^{(1)}(k) = 0.5X^{(1)}(k) + 0.5X^{(1)}(k-1) \tag{8}$$

5）根据 $X^{(1)}(k)$ 建立灰色微分方程如式（9）所示：

$$\frac{\mathrm{d}X^{(1)}}{\mathrm{d}t} + aX^{(1)} = u \tag{9}$$

6）采用 Lsm（Least square method）对参数 u 和 a 的值进行估计，得到的参数向量如下所示：

$$\begin{bmatrix} u \\ a \end{bmatrix} = (\boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{X}_N \tag{10}$$

在式（10）中，矩阵 $\boldsymbol{B}$ 和向量 $\boldsymbol{X}_N$ 可以表示为：

$$\boldsymbol{B} = \begin{bmatrix} -Z^{(1)}(2)1 \\ -Z^{(1)}(2)1 \\ \cdots \\ -Z^{(1)}(n)1 \end{bmatrix} \quad \boldsymbol{X}_N = \begin{bmatrix} X^{(0)}(2) \\ \cdots \\ X^{(0)}(n) \end{bmatrix} \tag{11}$$

7）根据累减法可以得到数据序列 $X^{(0)}$ 的在下一个周期的预测值，如下所示：

$$\overline{X^{(0)}}(k+1) = \overline{X^{(1)}}(k+1) - \overline{X^{(1)}}(k) \tag{12}$$

从算法 1 可以看出，每次根据存储的长度为 n 的历史负载信息，则可以判断该服务器池中的虚拟机在下一时刻的负载情况，从而及时地进行虚拟机的增加、删除和迁移。

3　基于预测和负载均衡的虚拟机迁移策略

ELB 算法中仅能根据当前负载的情况进行虚拟机的增加和删除，不能进行迁移，由于虚拟机的增加需要重新根据模板来创建实例，因此，需要耗费较多的资源，而虚拟机的迁移仅需将负载迁移到别的虚拟机上，因此，可以根据虚拟机负载的预测值来判断是进行迁移还是增加虚拟机。同时 ELB 算法中的另一个问题就是不能根据用户自己的需求选择最符合要求的模板，如在 CPU 满足条件的情况下，可能造成带宽和内存的浪费，因此，文中算法允许在服务器池中选择匹配度更高的虚拟机。

算法 2：基于预测和负载均衡的虚拟机迁移策略。

输入：虚拟机调度周期 T，服务器 i 的长度为 T/t 服务器的负载队列 Q_i，正在响应用户任务请求的服务器池队列 $poolist$，服务器池 $pool_j$ 的删除队列 $deleteQ_j$，每个服务器 i 对应的长度为 n 的历史负载信息 $lInf_i$，虚拟机增加、删除和迁移阈值分别为 $l_{\max}$、$l_{\min}$ 和 $l_{(\mathrm{mid})}$；

1）对于当前任务请求随机分配虚拟机 c，如果其任务负载数已满为 $|T/t|$，则根据式（1）计算其负载队列 Q_c 的平均负载 $AveL_c$，否则将 $AveL_c$ 置为 0；

2）如果 $AveL_c$ 不为 0，则查找虚拟机 c 其所属于的服务器池 $pool_c$，并将 $pool_c$ 加入其所属服务器池队列 $poolist$ 中；

3）根据式（2）计算服务器池队列 $poolist$ 中的每个服务器池的平均负载 $Avepoolh_i$：

4）根据 $Avepoolh_i$ 的大小对服务器池队列 $poolist$ 中的所有服务器池进行排序；

5）取出服务器池队列中平均负载最大的服务器池，即的队首元素 $poolh$，对该服务器池 $poolh$ 中的所有服务器运行算法 1 根据存储的当前虚拟机长度为 n 的历史负载信息 $lInf_i$ 来预测在下一个时间周期 T_{next} 的负载情况 l_{next}；

6）对于 $poolh$ 中的每个虚拟机：

（1）如果其预测的负载 l_{next} 大于预设的最大阈值 $l_{\max}$ 时，则当前的负载 l_{cur}，则将该虚拟机对应的 $deleteQ$ 中的 Num 个服务器的由删除状态更改为正常状态，否则将 n_s 个服务器全部修改为正常状态，并根据改进的 AMIs 镜像模板新建 $Num-n_s$ 个虚拟机，并将其加入到服务器池 $poolh$ 中，改进的 AMIs 镜像模板可以表示为：

$$ml=\left(\frac{al_c-u_c}{al_c},\frac{al_m-u_m}{al_m},\frac{al_b-u_b}{al_b}\right) \tag{12}$$

其中：(u_c,u_m,u_b) 表示用户需求的 CPU 资源、内存和带宽单位个数，而 (al_c,al_m,al_b) 表示改进的 AMIs 镜像模板，选择满足 $0.05\leqslant ml\leqslant 0.1$ 的模板 (al_c,al_m,al_b) 作为定制的改进模板，从而生成新的虚拟机。

（2）如果其预测的负载 l_{next} 大于预设的迁移阈值 $l_{(\mathrm{mid})}$ 但小于最大阀值 $l_{\max}$ 时，此时，需要进行虚拟机的负载迁移，即将负载均衡器发出迁移请求，负载均衡器在接到迁移请求后，选择当前服务器池中具有最小负载的虚拟机，并将其发送虚拟机迁移请求，当负载均衡器获得最小负载的虚拟机的请求允许消息后，向发送迁移请求的虚拟机发送迁移允许消息，此后，虚拟机将当前执行的任务的镜像通过负载均衡器转发到目标虚拟机。

（3）如果其预测的负载 l_{next} 大于预设的删除阈值 $l_{\min}$ 但小于迁移阀值 $l_{(\mathrm{mid})}$ 时，此时，不需要进行负载均衡；

（4）如果其预测的负载 l_{next} 小于预设的删除阈值 $l_{\min}$ 时，此时，将所有负载删除，并将负载对应的用户请求插入到请求等待队列中，将该虚拟机删除。

7）重复步骤 6），直到当前服务器池中的每个虚拟机均已实现负载平衡，此时转到步骤 5）选择负载最大的服务器池，继续进行虚拟机增加、删除或迁移，实现负载均衡。

4　仿真实验

4.1　实验环境

为了验证文中方法的可行性和优越性，采用 CloudSim 仿真环境下将文中方法与经典的负载均衡调度算法—ELB 算法以及文献［9］进行对比，参数设置如下：物理节点数量为 50，CPU 计算能力/（MI s^{-1}）为 2 000、3 000 和 6 000，内存/（GB）为 2、4 和 8，网络带宽/（MI s^{-1}）为 60、100、150，任务总数为 10 00 个，对 CloudSim 中的 DataCenterBroker 类进行修改，在类中每隔 5 s 随机产生新任务，任务个数与任务分配均为随机的，因此不能服务器池中的虚拟机在不同时刻均有不同的负载。

4.2　实验结果与分析

将 m 次实验的负载值作为数据，通过负载均衡因子来衡量算法所能实现的负载均衡程度，负载均衡因子可以定义为：

$$\varphi=\sqrt{\frac{\sum_{j=1}^{m}(LB_k-L\bar{B}_{jk})^2}{m-1}} \tag{13}$$

其中：LB_j 为虚拟机 r_j 在某时刻 k 的负载均衡值，$L\bar{B}_j$ 为云计算数据中心在某时刻 k 的平均负载。

每隔 10 s 对云计算数据中心的平均负载进行一次计算，运行实验 50 次，得到的实验结果如下所示：

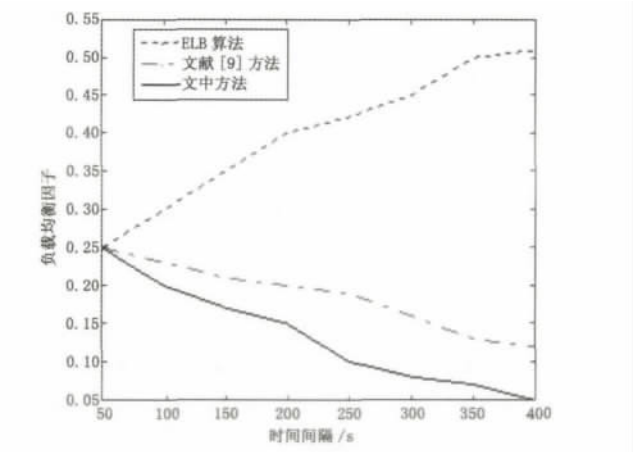

图 1　负载均衡效果比较

从图 1 可以看出，ELB 算法的负载均衡能力并不理想，在整个仿真期间均高于另外两种方法，且在 400 s 时，达到最高值 0.51，而另外两种方法随着仿真时间的增加，负载均衡效果反而更好，这是因为初始的任务随机分配导致虚拟机的负载不平衡，随着时间的进行，负载均衡算法能有效地根据各虚拟机负载的变化，动态的进行虚拟机的增加 h 和删除，因此，

随着仿真的进行负载均衡程度进一步得到改善，文中方法较文献［9］更优是因为文中方法还引入了虚拟机的迁移，文献［9］仅进行增加和删除，因此，负载均衡能力最好。

当计算出所有虚拟机的负载均衡因子后，可以根据下式计算负载均衡效率：

$$ET_k = \frac{load_0 - load_k}{T_k} \tag{14}$$

在式（14）中，$load_0$ 表示数据中心的负载均衡值，T_k 表示达到负载均衡值所需要的时间 $load_k$，式（14）可以看出，ET_k 的值越小，其效率越高。

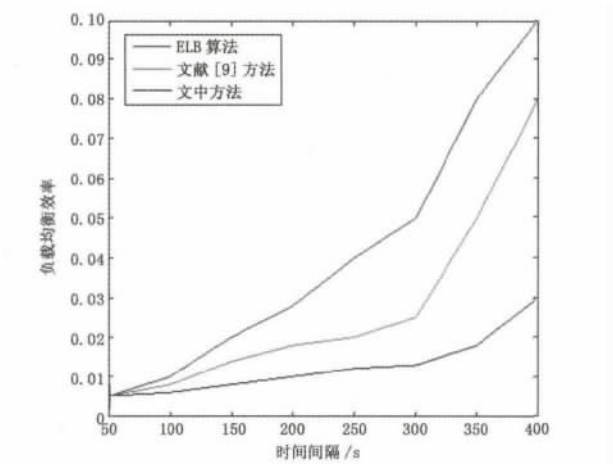

图 2 负载均衡效率

从图 2 可以看出，文中方法的负载均衡效率较高，远远高于 ELB 算法和文献［9］算法，文中方法的平均值 0.012 7，ELB 算法和文献［9］算法分别为 0.041 6 和 0.027 5，显然，文中方法的负载均衡效率最高，这是因为文中方法能通过任务的迁移有效地减少增加虚拟机造成的开销，同时通过定制虚拟机模板使得用户的请求均能得到及时响应，因此，具有较高的负载均衡效率。

5 结束语

为了实现云计算中心的负载均衡，设计了一个基于 GM（1，1）预测机制的虚拟机调度迁移策略。介绍了云计算中心负载均衡的原理并描述了 ELB 负载均衡算法，然后提出了基于 GM（1，1）的负载预测算法，能根据历史负载信息来实现下一时刻负载预测，从而提高用户任务的响应速度，在此基础上定义了能实现虚拟机增加、删除和迁移的负载均衡算法，能有效地实现用户任务请求的均衡调度，具有很强的可行性。

参考文献：

［1］Armbrust M，Fox A，Griffith R，et al. Above the clouds：a Berkeley view of cloud comuputing，UCB/EECS－2009－28［R］. Springfield，USA：University of California，Berkely. Electrical engineering and computing science department，2009.

［2］庄 威，桂小林，林建材，等. 云环境下基于多属性层次分析的虚拟机部署与调度策略［J］. 西安交通大学学报，2013，2（47）：28－32.

［3］Goyal A，Bonchi F，Lakshmanan L V. Learning influence probabilities in social networks［A］. proceedings of the third acm international conference on web search and data mining［C］. New York，USA：Acm，2010：241－250.

［4］Armbrust M，Fox A，Griffith R，et al. A View of Cloud Computing［J］. Communications of the ACM（S0001－0782），2010，53（4）：50－58.

［5］Brian H. Cloud Computing［J］. Communications of the acm（s0001－0782），2008，51（7）：9－11.

［6］Rodrigo N Calheiros，Rajiv Ranjan，Anton Beloglazov，et al. Cloudsim：a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms［J］. Software：Practice and Experience，2011，41（1）：23－50.

［7］Jain R，Rao B. Application of Ahp Tool for Decision Marking of Choice of Technology for Extraction of Anti－cancer Bioactive Compounds of Plant Origin［J］. International Journal of the Analytic Hierarchy Process，2013，5（1）.

［8］李鹏伟，葛文英. 云计算环境下虚拟机动态部署研究［J］. 计算机测量与控制，2013，21（5）：1374－1376.

［9］吴和生，王崇骏，谢俊元. TeraPELB：云计算中基于预测的弹性负载均衡算法［J］. 系统仿真学报，2013，25（8）：1751－1765.

［10］龚素文，艾浩军，袁远明. 基于迁移技术的云资源动态调度策略研究［J］. 计算机工程与应用. 2014，50（5）：51－54.

［11］胡元元，林 浒，李鸿彬. IaaS 云中最小迁移代价的虚拟机放置算法［J］. 小型微型计算机系统，2014，35（4）.

［12］孙冬冬，柳 青，武旖旎. 面向负载均衡的自主式虚拟机动态迁移框架［J］. 计算机科学，2014，41（4）：80－85.

（上接第 3746 页）

系统综合的关键，并逐渐会成为小卫星综合电子计算机设计的方向。本文提出了使用双口 RAM 来实现多 CPU 之间数据共享和传输的方案，并就 CPU 之间数据竞争和仲裁提出了适用于小卫星任务特点的应用用例，实验结果表明，该系统具有极高的实时性和数据处理能力。后续，小卫星面临智能化、集成化和综合化的新需求，还需要在空间环境适应性设计、可扩展性设计[7]和多 CPU 数据竞争算法优化设计等几个方面进行研究，使得双口 RAM 在多 CPU 架构的综合电子计算机实现真正高效的工程应用。

参考文献：

［1］王九龙. 卫星综合电子系统现状和发展建议［J］. 航天器工程，2007，16（5）：68－73.

［2］Franz Muehlegg，Alfred Schuetz. A highly flexible dual－port－RAM compiler［J］. IEEE Euro ASIC，1990，（6）：277－281.

［3］Integrated Device Technology. IDT7028L CMOS Dual－Port RAM datasheet［Z］. 2009.

［4］Jonathan W. Valvano. 嵌入式微计算机系统［M］. 北京：机械工业出版社，2003.

［5］范庆辉，张 蕾，阳富民. 嵌入式系统双口存储器应用研究［J］. 计算机工程与设计，2008，29（12）：3085－3087.

［6］韩 钧，康 怡. 双口 RAM 在 DSP 与单片机数据通信中的应用［J］. 电力系统通信，2006，27（8）：56－58.

［7］童时中. 模块化原理设计方法及应用［M］. 北京：中国标准出版社，2005.